Sustainable Operations and Computers 5 (2024) 15–28

Contents lists available at ScienceDirect

Sustainable Operations and Computers

journal homepage:
http://www.keaipublishing.com/en/journals/sustainable-operations-and-computers/

Efficient algorithms for electric vehicles' min-max routing problem

Seyed Sajjad Fazeli [a], Saravanan Venkatachalam [a,*], Jonathon M. Smereka [b]

[a] Department of Industrial and Systems Engineering, Wayne State University, Detroit, MI, USA
[b] Researcher within the Ground Vehicle Robotics (GVR) team, U.S. Army CCDC Ground Vehicle Systems Center (GVSC), Warren, MI, USA

ARTICLE INFO

Keywords:
Electric vehicles
Routing
Charging station
Hybrid heuristic
Variable neighborhood search

ABSTRACT

An increase in greenhouse gases emission from the transportation sector has led companies and the government to elevate and support the production of electric vehicles (EV). With recent developments in urbanization and e-commerce, transportation companies are replacing their conventional fleet with EVs to strengthen the efforts for sustainable and environment-friendly operations. However, deploying a fleet of EVs asks for efficient routing and recharging strategies to alleviate their limited range and mitigate the battery degradation rate. In this work, a fleet of electric vehicles is considered for transportation and logistic capabilities with limited battery capacity and scarce charging station availability. We introduce a min-max electric vehicle routing problem (MEVRP) where the maximum distance traveled by any EV is minimized while considering charging stations for recharging. We propose an efficient branch and cut framework and a three-phase hybrid heuristic algorithm that can efficiently solve a variety of instances. Extensive computational results and sensitivity analyses are performed to corroborate the efficiency of the proposed approach, both quantitatively and qualitatively. Finally a data-driven simulation implemented with the robot operating system (ROS) middleware are performed to corroborate the efficiency of the proposed approach, both quantitatively and qualitatively.

1. Introduction

Global warming has been primarily linked to human activities which release greenhouse gases [1]. Among those activities, the transportation sector causes the largest share (about 28%) of greenhouse gas emissions, which mainly originate from fossil fuel burner vehicles [2,3]. In an effort to offset carbon emissions from fossil fuel burning vehicles, priority to transform the transportation systems by driving new technological innovations in vehicles [4] is causing electric vehicles (EVs) to become rapidly important for many automotive companies. Many countries have offered incentives to accelerate the adoption of EVs to increase the EV share in future vehicle fleets [5].

It is important for EVs to choose energy-efficient routes and find the best locations for recharging during their itineraries. Transportation network companies (e.g. Lyft and Uber), and logistic companies have (e.g., FedEx and UPS) already started to operate a fleet of EVs in their business for last mile deliveries [6]. Adopting EVs also brings new challenges. One of the main operational challenges for EVs in transport applications is their limited range and the availability of charging stations (CS) [7–9]. It is estimated that half of the US population lives in areas with fewer than 90 charging infrastructures per million people [10,11]. To successfully employ EVs, we need strategies that can alleviate the range and recharging limitations.

The electric vehicle routing problem with limited range and number of CSs presented in this work, falls into the category of green vehicle routing problem (GVRP). The GVRP embraces a broad and extensive class of problems considering environmental issues as well as finding the best possible routes for vehicles. The GVRP research can be broadly divided into two categories: 1) minimize the fuel consumption while considering loading weights [12]; and 2) replace the conventional vehicles with alternative fuel vehicles (AFV) [8,13–17]. This research focuses on AFVs, hence we briefly review the related literature on routing strategies for AFVs. An initial work was done by [18], where the authors developed a mixed integer programming (MIP) formulation and a genetic algorithm (GA) to overcome the range limitation of AFVs and shortage of refueling locations. Authors in [8] introduced the electric vehicle routing problem (EVRP) with time windows and charging stations with the limited freight capacity for the vehicles. They developed a hybrid meta-heuristic by integrating variable neighborhood and Tabu search. For single unmanned aerial vehicles' routing problem, authors in [13] proposed a novel approach based on an approximation algorithm combined with a heuristic method. Later, the authors extended their work to multiple vehicles by applying the Voronoi algorithm as the construction phase, and 2-opt, 3-opt variable neighborhood searches in the improvement phase [19].

* Corresponding author at Distribution A. Approved for public release distribution is unlimited. OPSEC # 4492.
E-mail addresses: sajjad.fazeli@wayne.edu (S.S. Fazeli), saravanan.v@wayne.edu (S. Venkatachalam), jonathon.m.smereka.civ@mail.mil (J.M. Smereka).

https://doi.org/10.1016/j.susoc.2023.07.002
Received 18 February 2023; Received in revised form 29 May 2023; Accepted 24 July 2023
Available online 28 July 2023

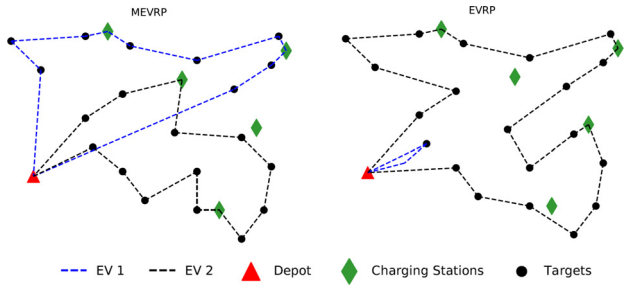

Fig. 1. A feasible tour for two EVs visiting all the targets while visiting some charging stations for recharging: MEVRP (left) vs EVRP (right).

In this work, the MEVRP is defined as follows: given a set of EVs which are initially stationed at a depot, a set of targets and a set of CSs, the goal is to visit each target exactly once by any EV and return to the depot while no EV runs out of charge while they travel their respective routes. It is assumed that all EVs will be fully charged at any visited CS, and the fuel consumption rate is linearly proportioned with traveled distance. The objective of this problem is to minimize the maximum distance traveled by any EV instead of the total distance, which is conventional in VRP. The MEVRP is fundamentally different from the EVRP. An optimal solution in MEVRP assigns routes to all EVs such that none of the EVs has a longer route. This results in a balanced distribution of loads and fair and equitable utilization of the EVs, which can decrease the rates of battery degradation in EVs [20]. Refer to Fig. 1 for an illustration for MEVRP and EVRP routes for EVs. In Fig. 1, EV1 visits most of the targets and travels a lot of distance compared to EV2 with min-sum. However, using MEVRP, with a nominal increase in the overall distance, the two EVs travel almost the same amount of distance. In addition, the number of recharging of EVs in more evenly distributed in the min-max comparing to min-sum.

The MEVRP is also of interest when the time to visit every target from the base depot is more important than the total traveled distance in applications like surveillance, emergency and disaster management [21], intelligence, and reconnaissance [22–24], and multi-robot path planning problems. In energy-efficient multi-robot path planning, the goal is to obtain optimal paths for each robot while avoiding obstacles in the presence of recharging points [25]. Additionally, the min-max provides a fair and equitable utilization for the resources, and the maximum wait time for any target will be less compared to a solution from min-sum. This is particularly vital if the EVs are used to transport people, and the targets are considered as stops in the EVs' routes. There are quite a few studies in the min-max VRP which differ by solution methodologies. These problems are often solved by heuristic methods with multiple phases for constructing the initial solution(s), and subsequently, improving them. Methodologies differ based on the construction of initial solution and the number of base depots. The author in [26] considered a single depot min-max vehicle routing problem (SDVRP) where they used an ant colony system as well as a random approach to assign targets to the vehicles. A 3-opt method is used to improve solutions. The work in [27] considered a capacitated SDVRP where they generated initial solutions by a parallel greedy insertion method and improved them by an adaptive variable neighborhood search (VNS). In multi-depot min-max VRP, usually, the authors use partitioning techniques to transfer the min-max MDVRP to a set of SDVRPs, and solve each SDVRP separately [28–32].

To the best of our knowledge, this study is the first attempt to formulate and solve the min-max version of MEVRP where range limitation is defined for each EV with a set of charging stations. We propose an efficient MIP formulation to solve small-scale instances. For large-scale instances, we develop a hybrid heuristic algorithm (HHA) where we obtain initial solution using an integer programming model and a heuristic, and subsequently, Variable Neighborhood Search (VNS) and genetic algorithm are used to improve the solutions along with novel feasibility methods. Extensive computational experiments evaluate all the proposed approaches.

The contributions of this study include the following: 1) an efficient MIP formulation for the MEVRP; 2) an HHA with an embedded feasibility method for large-scale instances and extensive computational experiments to quantify the efficacy of the proposed approach; 3) computational experiments for MIP formulation using branch and cut algorithm; 4) a sensitivity analysis to investigate the aspects of solutions from EVRP and MEVRP; 5) a data-driven simulation study implemented with the robot operating system ROS middleware.

The remainder of this paper is organized as follows: Section 2 provides a mathematical formulation of the problem along with a subsequent reformulation. Section 3 introduces the solution methodologies where we present exact and heuristic methods to solve small and large-scale instances, respectively. Section 4 presents extensive computational experiments and sensitivity analysis. Finally, Section 6 provides concluding remarks.

2. Model formulation

2.1. Problem definition

We define T as a set of targets, and $\bar{D}$ as a set of CSs. Define $D = \bar{D} \cup d_0$, be a set of CSs, including a depot d_0 where m EVs are initially stationed, and each EV is charged to its battery capacity. The MEVRP is defined on a directed graph with a set of vertices V and a set of edges E as $G = (V, E)$ where $V = T \cup D$. We assume that the graph G does not contain any self-loop. Each edge $(i, j) \in E$ is associated with a non-negative cost c_{ij} between vertices i and j. It is assumed to be directly proportional to the energy consumption $c_{i,j} = \epsilon \cdot f_{ij}^m$, where f_{ij}^m is the amount of energy consumption by traveling from i to j and ϵ is a constant denoting the energy consumption rate of EV m. It is also assumed that both the distances and the charge costs satisfy the triangle inequality, e.g., $\forall i, j, k \in V$, $f_{ij}^m + f_{jk}^m \geq f_{ik}^m$. Also, let F_m denote the maximum charging capacity of any EV m. The objective of the model is to find a route for each EV starting and ending at the base depot such that: each target is visited at least once by an EV; no EV runs out of charge during the trip; and maximum distance traveled by an EV is minimized. The objective functions of MEVRP is represented as follows: Min $\left(\underset{\forall m \in M}{\text{Max}} \sum_{(i,j)\in E} c_{ij} x_{ij}^m\right)$, and represented as a linear function in the following mathematical model.

2.2. Notation

- Sets
  - T: Set of targets, indexed as $t \in T$.
  - $\bar{D}$: Set of charging stations, indexed as $d \in \bar{D}$.
  - D: Set of charging stations and base depot, $D = \bar{D} \cup \{d_0\}$.
  - V: Set of all vertices in the graph, including all targets, CSs and base depot, $V = T \cup D$.
  - E: Set of all edges connecting any two vertices without any self-loop, $(i, j) \in E$ and $i, j \in V$.
  - S : Subset of targets and a depot in V, $S \subset V$, $\sigma^+(S) = \{(i, j) \in E : i \in S, j \notin S\}$.
  - M: Set of EVs which are initially stationed at base depot d_0, indexed by $m \in M$.
- Model parameters
  - c_{ij}: Cost of traversing an edge $(i, j) \in E$.
  - f_{ij}^m: Amount of energy consumption of EV m by traveling from node i to j with $i, j \in V$.
  - F_m: Battery capacity of EV m.
  - q: A large constant, set as number of targets.
- Decision variables
  - x_{ij}^m: 1 if the edge (i, j) is traversed by an EV m, and 0 otherwise;

- z_{ij}^m: Flow variable associated with each edge $(i,j) \in E$ indicating the amount of charge consumed by EV m.
- y_d^m: 1 if the CS $d \in D$ is visited by any EV m, and 0 otherwise.
- w: Maximum traveled distance among the EVs.

2.3. MEVRP model

$$\text{Min } w \tag{1}$$

s.t.

$$w \geq \sum_{(i,j)\in E} c_{ij} x_{ij}^m \quad \forall m \in M, \tag{2}$$

$$\sum_{i\in V} x_{di}^m = \sum_{i\in V} x_{id}^m \quad \forall d \in \bar{D}, m \in M, \tag{3}$$

$$\sum_{i\in V} x_{di}^m \leq q.y_d^m \quad \forall d \in \bar{D}, m \in M, \tag{4}$$

$$\sum_{i\in V} x_{id_0}^m = 1, \sum_{i\in V} x_{d_0 i}^m = 1 \quad \forall m \in M, \tag{5}$$

$$\sum_{i\in V}\sum_{m\in M} x_{ij}^m = 1, \sum_{i\in V}\sum_{m\in M} x_{ji}^m = 1 \quad \forall j \in T, \tag{6}$$

$$x^m\left(\sigma^+(S)\right) \geq y_d^m \quad \forall d \in S \cap \bar{D}, S \subset V \setminus \{d_0\} : S \cap \bar{D} \neq \emptyset, m \in M, \tag{7}$$

$$\sum_{j\in V} z_{ij}^m - \sum_{j\in V} z_{ji}^m = \sum_{j\in V} f_{ij}^m x_{ij}^m \quad \forall i \in T, m \in M, \tag{8}$$

$$z_{ij}^m \leq F_m x_{ij}^m \quad \forall (i,j) \in E, m \in M, \tag{9}$$

$$z_{di}^m = f_{di}^m x_{di}^m \quad \forall i \in T, d \in D, m \in M, \tag{10}$$

$$x_{ij}^m \in \{0,1\}, z_{ij}^m \geq 0 \quad \forall (i,j) \in E, m \in M, \tag{11}$$

$$y_d^m \in \{0,1\} \quad \forall d \in \bar{D}, m \in M. \tag{12}$$

The objective function (1) minimizes the maximum distance traveled by any EV. Constraints (2) represents the upper-bound for the traveled distance of any EV m using the continuous variable w. Constraints (3) ensure the in-degree and out-degree of any EV using CS d to be equal. Constraints (4) force $y_d^m = 1$ if EV m visits CS d. Constraints (5) ensure that EVs start and end their trip from the base depot d_0. Constraints (6) guarantee that each target should be visited exactly once and by one EV. Connectivity of any feasible solution is guaranteed by constraints (7). Constraints (8) introduce the flow variable z_{ij}^m for each edge $(i,j) \in E$ and also removes the sub-tours. Constraints (9) and (10) ensure that no EV runs out of charge during its trip. Finally, constraints (11) and (12) define the restrictions for the decision variables.

Proposition 2.1. *The following constraints are valid LP-relaxation of constraints (4) and (12):*

$$x_{di}^m \leq y_d^m \quad \forall i \in T \cup \{d_0\}\ d \in \bar{D}, m \in M, \tag{13}$$

$$0 \leq y_d^m \leq 1 \quad \forall d \in \bar{D}, m \in M. \tag{14}$$

Proof. Since constraints (13) ensure that EV m can use a charging station d only if $y_d^m = 1$, hence they are valid constraints for MEVRP. Besides, they force the value of y_d^m to be either 0 or 1 as $x_{di}^m \in \{0,1\}$ for any $i \in T \cup \{0\}$. Therefore, the binary restriction on variables y_d^m are relaxed in (14). The proof is based on [7] □

3. Methodology and algorithm development

3.1. Branch and cut algorithm

In this section, we describe the main components of a branch-and-cut algorithm used to optimally solve the formulation presented in Section 2. Majority of previous studies use a set of dummy binary variables for CSs to maintain the connectivity of the EV tours ([8,18,33]), and the connectivities should be determined prior to the optimization by the users. This type of formulation can significantly increase the computational effort due to poor LP-relaxation. However, our formulation contains constraints (7) to guarantee the connectivity of any feasible solution without using dummy binary variables. But, the number of such constraints is exponential, and it may not be computationally efficient to consider all these constraints in advance while using an off-the-shelf solver. To address this issue, we relax the constraints (7) from the formulation. Whenever there is a feasible integer solution, we check if any of the constraints (7) are violated. If so, we add the corresponding constraint and continue solving the problem. It has been observed that this process is computationally efficient for a variety of the VRP problems [34,35].

Now, we describe the details about the algorithm used to find a constraint (7) that is violated for a given integer feasible solution to the relaxed problem. For every EV $m \in M$, a violated constraint (7) can be denoted by a subset of vertices $S \subset V \setminus \{d_0\}$ such that $S \cap \bar{D} \neq \emptyset$; and $x^m(\sigma^+(S)) < y_d^m$ for every $d \in S \cap \bar{D}$ and for every EV. We construct an auxiliary graph $G' = (V', E')$ for any feasible solution where $V' = T \cup d_0 \cup \{d \in \bar{D} : y_d^m = 1\}$, and $E' = \{(i,j) \in E : x_{ij}^m = 1\}$. We then find the strongly connected components (SCC) of this graph. Every SCC sub-graph which does not contain the base depot d_0 violates the constraint (7). Hence, we add all these constraints for any feasible integer solution until we reach optimality. To implement this algorithm within the branch and cut framework, we use the *Callback* feature provided by most of the commercial solvers like Gurobi [36]. Although the branch-and-cut can find an optimal solution, the MEVRP is an extension of VRP problem and it is NP-hard [18]. Thus, to circumvent computational challenges for large-scale instances, we develop a hybrid heuristic method in the next section.

3.2. HHA method

To solve MEVRP using a heuristic algorithm, we have three major challenges: 1) assigning targets to the EVs; 2) finding the best route for each EV; and 3) maintaining feasibility such that each EV does not run out of fuel. Each of these is complex, hence a naive heuristic may not be sufficient. Therefore, we implemented hybrid heuristic algorithm by integrating an linear programming model, VNS, genetic algorithm and multiple heuristics in three different phases to produce high-quality solutions [37].

The flowchart of HHA for MEVRP is shown in Fig. 2. The algorithm initializes by computing a modified traveling cost matrix for each EV to consider the charging limitation of EVs. Subsequently, a linear programming (LP) relaxation of an assignment problem is solved to assign the targets to the EVs. Then, an optimal or a sub-optimal travelling salesman problem (TSP) tour for the EVs is determined by the Lin-Kernigan-Helgaun heuristic [38]. In the next step, the feasibility of each route in terms of range limitation before charging is checked. On every infeasible route, a novel heuristic is applied to find CSs for recharging and the distance traveled by each EV is recalculated, and an initial solution is obtained. For a pool of high-quality solutions, an iterative VNS procedure is implemented, and uses the initial solution as the incumbent. Three different insertions and a swap operation are used to improve the incumbent solution. At each iteration, a new solution is generated and compared to the incumbent. If the new solution is better than the incumbent, it is considered as the 'new' incumbent and added to a pool. Also, if a new solution is not better but has a cost relatively closer compared

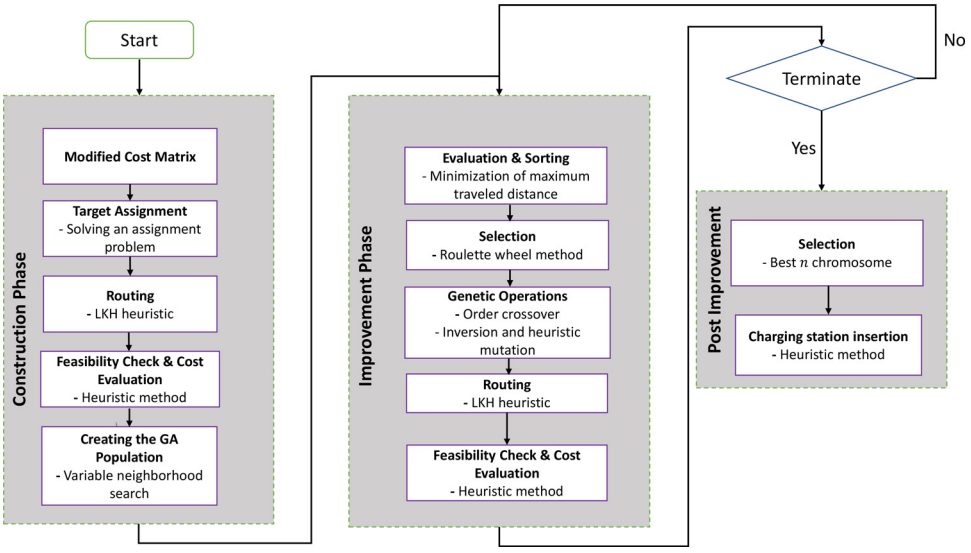

Fig. 2. Flowchart of the HHA approach including the three main phases: construction, improvement and post improvement.

to the incumbent, it is considered as a potentially good solution and added to the pool. This process is repeated until either no improvement is found or the maximum number of iterations is reached.

Once the pool is filled with high-quality solutions using VNS, GA parameters such as the iteration number, population size, crossover rate, mutation rate, stopping criteria are initialized. The solutions in the pool represent GA chromosomes, and the fitness value of chromosomes is considered as the maximum distance traveled by any EV. The chromosomes are sorted based on their fitness value and the ones with the higher fitness values are eliminated based on fixed population size. During the improvement phase, through a roulette wheel selection operation, some chromosomes are selected for the GA operations. The GA operations such as crossover and mutation are performed to generate new solutions (offsprings). The routing and feasibility check are performed again on the offsprings. The fitness value of the feasible offsprings is measured and compared to other chromosomes. These steps constitute an iteration, and then the roulette wheel selection is applied again to begin the next iteration. The HHA is terminated whenever a stopping criterion is met. In the post improvement, a heuristic is used on some of the best chromosomes to further improve the solution from the improvement phase. Steps of HHA are explained in the subsequent sections.

3.2.1. Construction phase

The goal of construction phase is to produce high-quality feasible solutions as initial solutions for the MEVRP. A series of steps are followed, and the details are elaborated in the following sections. *Path representation* To encode the solution of MEVRP problem, we use path representation in a way that targets are listed based on the order in which they are visited. Suppose that there are 10 targets and numbered from t_5 to t_{14}. In order to form a chromosome, we generate a path where targets are randomly placed in each gene of the chromosome. A sample chromosome of MEVRP problem is as follows:

t_{10}	t_7	t_{11}	t_{12}	t_9	t_5	t_6	t_8	t_{13}	t_{14}

Note that each chromosome contains $|M|$ string, each related to an EV. *Modified cost matrix* In this step, considering the range limitation of EVs, a new traveling cost based on [39] is computed. The new cost matrix considers the additional charge that an EV may need to visit the available CSs as it traverses between two targets. The maximum charge remaining at the EV's battery when it visits a target i is denoted as $F - f_{d_i}^{\min}$, where $f_{d_i}^{\min}$ denotes the amount of fuel an EV requires to reach a CS with the minimum distance from target i. This ensures that in any feasible tour, an EV must have an option of recharging at the nearest CS to continue visiting other targets when it reaches a target i. Hence, an EV can directly travel from target i to target j if and only if $f_{ij} \leq F - f_{d_i}^{\min} - f_{d_j}^{\min}$. If the EV is unable to directly travel from target i to j, an auxiliary graph $G^* = (V^*, E^*)$ is created where $V^* = \bar{D} \cup \{i, j\}$. Any edge that can satisfy the fuel constraint will be added to the auxiliary graph. The following three sets of edges to the graph:

$$E^* =: \begin{cases} (i,d): & \text{if } f_{id} \leq F - f_{d_i}^{min}, \forall d \in \bar{D}, \\ \cup(d,j): & \text{if } f_{dj} \leq F - f_{d_j}^{min}, \forall d \in \bar{D}, \\ \cup(d_k, d_{k'}): & \text{if } f_{d_k d_{k'}} \leq F, \forall d_k, d_{k'} \in \bar{D}. \end{cases}$$

For every EV, the cost (length) of the edges in the graph E^* is calculated by finding the shortest path between any two nodes using Dijkstra's Algorithm [40]. The modified cost matrix is created by replacing the new computed costs with old costs in the original cost matrix. Let the modified cost vector be denoted as c'.

Target assignment and routing To initially assign targets to the EVs, we use a LP-based Load Balancing Heuristic (LLBH) as suggested in [30] with some modifications. The LLBH assumes that the distance traveled by the EVs should almost be the same, and designed for multi-depot problems. Hence, in a network where targets are uniformly distributed, EVs visit nearly the same number of targets. For the sake of compatibility, we perturb the location of EV in the base depot to create multiple depots. We uniformly place the EVs on a small circle around the base depot as shown in Fig. 3. Given a set of K base depots, indexed by i where one EV is stationed at each of them, and set of T targets, indexed by j, the LLBH solves the relaxation of the following assignment problem where u_{ij} is 1 if target j is assigned to the depot i, and 0 otherwise. Also, c'_{ij} indicates the modified cost matrix.

$$\mathbf{P}\text{: Min} \sum_{i \in K} \sum_{j \in T} c'_{ij} u_{ij} \tag{15}$$

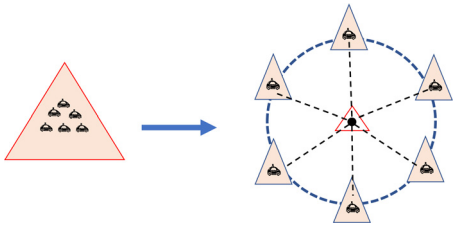

Fig. 3. Conversion of single depot to multi-depots using perturbation method.

s.t.

$$\sum_{i \in K} u_{ij} = 1 \qquad \forall j \in T, \tag{16}$$

$$\sum_{j \in T} u_{ij} = \lfloor \frac{|T|}{K} \rfloor \qquad \forall i \in K, \tag{17}$$

$$u_{ij} \in \{0, 1\} \quad \forall i \in K, j \in T. \tag{18}$$

Constraints (16) assign targets to the 'K' copies of depots made around the base depot d_0 where each depot contains one EV. Constraints (17) determine the number of targets that each EV should visit. For the cases where $\frac{|T|}{K}$ is fractional, as per [30], we assume that $|T| = pK + r$, with $p, r \in \mathbb{Z}^+$ and r is residue. The extra r targets are assigned to the EVs based on a saving technique described in the next section. After the initial assignment of targets to EVs by solving **P**, the auxiliary depots are removed. Then, we add the base depot (d_0) to the beginning and end of each route which result in a group of strings. Now, we solve a single EV routing problem for every set of targets assigned to each EV including the base depot using the Lin-Kernigan-Helgaun (LKH). LKH is considered as one of the best algorithms for solving single vehicle routing problem without recharging constraints.

Feasibility check The route found by LKH heuristic for each EV could be infeasible due to the exclusion of range constraint (10). Unlike the other approaches in literature where the infeasible routes are dismissed or penalized, a novel feasibility approach is used to convert them to feasible routes. This method is specifically beneficial when number of CSs are limited. Hence, this step is performed to attain feasibility for the tours. We calculate the charge consumption of the EVs as they traverse along their route. Whenever the charge consumption exceeds the battery capacity, we stop at the last target (e.g. t_i) that EV visited before recharging. At this step, a CS should be selected for recharging. The algorithm developed by [33] considers the minimum of $dist(t_i, d_k)\ \forall k \in \bar{D}$, where '$dist$' represent distances based on the parameter f. However, in this work, we choose the minimum of $dist(t_{i+1}, d_k)\ \forall k \in \bar{D}$. The advantage of this approach is that the EV will have more charge to complete the rest of the trip, and also has less necessity for recharging. Consequently, this may reduce the overall distance. However, if the EV does not have sufficient charge to reach d_k, then the minimum of $dist(t_i, d_{k'})$ $\forall k' \in \bar{D} \setminus \{k\}$ is selected for recharging. If visiting $d_{k'}$ is also not possible, then "backward" move is performed to reach the previous edge (t_{i-1}, t_i) and perform the similar evaluation. A route is infeasible if one of the following circumstances occurs: 1) backward moves resulted in visiting the base depot; 2) all the available CSs in one edge are visited and yet it is impossible to move to the next edge; or 3) exiting from a target is impossible due to insufficient charge, and also the closet CS to the target is visited in the previous edge.

The feasibility check procedure is given in Algorithm 1. In this algorithm, a MEVRP route is defined as a sequence of targets $(t_0, t_1, t_2, \dots, t_{i-1}, t_i, t_{i+1}, \dots, t_{n+1})$ where the t_0 and t_{n+1} represent the depot. We also define $m_{(t_i, t_{i+1})} = 1$ if the edge $(i, i+1)$ is traversed by an EV. We consider $c^{\min}_{d_i}$ for the CS d which has the minimum distance to the target i. Also, $d^k_{(v_i, v_{i+1})} = 1$ if the eligible CS $d^k, \forall k \in \bar{D}$ is visited by the EV, and location of the EV is referred as loc.

The feasibility procedure is applied on every route to create a feasible solution along with its corresponding cost for the MEVRP problem. This solution x is considered as the incumbent solution $x_{inc} = x$. *Variable Neighborhood Search (VNS)*

To create a pool of high-quality solutions for the GA, a VNS-based heuristic is used. The VNS framework is based on a systematic change of a neighborhood integrated with a local search [41]. A local search starts with an initial solution x_{inc} and looks for a descent direction from x within a neighborhood $N(x)$. The algorithm stops if there is no further improvement. In this problem, four neighborhoods are used, including

Algorithm 1 Feasibility Check.

Initialization:
Set: *Status* ← Feasible, *BM* ← False, *FM* ← True,
F ← Battery Capacity, $i \leftarrow 0$, $loc \leftarrow t_i$
while *any* $m_{(t_i, t_{i+1})} \neq 1$:
if *Status* = *Infeasible*
break
else *do FM*
$i++$
while *FM* = *False*
do BM

Forward Move (FM):
if: $z \geq f_{(t_i, t_{i+1})}$:
loc ← t_{i+1}, $z- = f_{(t_i, t_{i+1})}$, $m_{(t_i, t_{i+1})} = 1$
Return True
elif: $z \geq f_{(t_i, c^{\min}_{d_{i+1}})}$ *and* $F \geq f_{(c^{\min}_{d_{i+1}}, t_i+1)}$ *and* $m_{(t_i, t_{i+1})} \neq 1$:
loc ← t_{i+1}, $z = F - f_{(c^{\min}_{d_{i+1}}, j)}$, $m_{(t_i, t_{i+1})} = 1$, $d^k_{(t_i, t_{i+1})} = 1$
Return = True
elif: $z \geq f_{(i, c^{\min}_{d_i})}$ *and* $F \geq f_{(t_i, c^{\min}_{d_i})}$ *and* $m_{(t_i, t_{i+1})} \neq 1$:
loc ← t_{i+1}, $z = F - f_{(c^{\min}_{d_i}, t_{i+1})}$, $m_{(t_i, t_{i+1})} = 1$, $d^k_{(t_i, t_{i+1})} = 1$
Return True
else: Return False

Backward Move (BM):
if: loc = t_0 **or** $d^k_{(t_i, t_{i+1})} = 1, \forall k \in \bar{D}$
Status ← Infeasible
Stop
else:
loc ← t_{i-1}, $z+ = f_{(t_{i-1}, t_i)}$, $m_{(t_{i-1}, t_i)} = 0$, $d^k_{(t_{i-1}, t_i)} = 0$,
do **FM**

one, two, and three insertions referred as $N_1(x)$, $N_2(x)$ and $N_3(x)$, respectively, and swap operation as $N_4(x)$. In the insertion procedure, the initial solution is improved by removing a target from the longest route and inserting to another route. Based on initial solution obtained in the previous section, a new solution x' from $N_1(x)$ is generated. In order to select the target i for removal, the savings due to removing each target from the longest route is calculated as follows:

$$dist(t_j, t_i) + dist(t_i, t_k) - dist(t_j, t_k), \tag{19}$$

where t_j and t_k are the preceding and succeeding targets in the same route. Target with the highest savings from the longest route is inserted into another route.

In the next step, the route for insertion of target is selected. The insertion cost of the selected target in other routes is calculated, and the route with the minimum insertion cost is selected. The insertion cost of each route is calculated using (19) where t_i is inserted between any two consecutive targets t_j and t_k from the route. If $f(x') < f(x_{inc})$ then $x_{inc} = x'$, where $f(.)$ is the objective function represented in (1). Then, we apply the LKH and feasibility check on the new solution The procedure continues with a new solution from $N_1(x')$, otherwise, the next target with the highest savings is selected. If there is no improvement by removing any target from the longest route and inserting into another route, a new solution from $N_2(x)$ is used for next iteration.

Whenever there is a better solution from $N_2(x)$, the iterations starts over from 1-insertion neighborhood. Otherwise, another target is selected for the next iteration. If there is no improvement in the 2-insertion neighborhood, $N_3(x)$ is used and continued as before. Apart from inser-

tions, a swap operation $N_4(x)$ is also used by randomly selecting a target from the longest route and exchanging it with any target from the route with the lowest insertion cost to further improve the solution. The swap operation is activated between any insertion operations (among $N_1(x)$, $N_2(x)$, and $N_3(x)$) and repeated for a predetermined number of iterations ($l_s^{\max}$). The procedure is terminated if the predefined number of non-improvement iterations is reached.

The VNS algorithm procedure is summarized in Algorithm 2.

Algorithm 2 Variable Neighborhood Search .

Initialization:
$l, k \leftarrow 0$, $x_{inc} \leftarrow x^0$, $N_1 \leftarrow$ True, $N_2 \leftarrow$ False,
$N_3 \leftarrow$ False, $i \leftarrow 1$
while $k \leq l_i^{\max}$:
  while N_i = True
    $k \leftarrow k + 1$
    select $x' \in N_i(x_{inc})$
    if $f(x') \leq f(x_{inc})$:
      $x_{inc} = x'$, $i \leftarrow 1$, $l, k \leftarrow 0$
    if $\nexists\, x \in \{x' | x' \in N_i(x'), f(x') \leq f(x_{inc})\}$:
      $N_i \leftarrow$ False
      $i \leftarrow i + 1$, $N_i \leftarrow$ True
      while $l < l_s^{\max}$
       do swap,
       if $f(x') \leq f(x_{inc})$:
       $x_{inc} = x'$, $i \leftarrow 1$, $l, k \leftarrow 0$

During the search process, every new solution is added to the GA pool. Inspired from simulated annealing, potential good solutions are stored. A solution is potentially good if $\frac{f(x')-f(x_{inc})}{f(x_{inc})} \leq s$, where $f(x')$ and $f(x_{inc})$ are the costs of the potentially good and incumbent solutions, and s is a relatively small parameter (e.g., s=0.15 in our implementation). In the next step, we apply the GA operations on the pool of high quality solutions.

3.3. Improvement phase

3.3.1. Chromosome selection

Chromosome selection significantly affects the GA's convergence. The roulette wheel selection was first introduced by [42] to select the chromosomes for GA operations. Each section of the roulette wheel is assigned to a chromosome based on the magnitude of its fitness value. The fitness value of each chromosome is based on objective function value given in (1). The fitness values of the chromosomes determine their chance of being selected.

3.3.2. Crossover and mutation

Two approaches are used to diversify the solution space. The first approach is 'order crossover' introduced in [43] to sample and combine selections from different potential solutions. In this approach, two offsprings are generated in each iteration by choosing a sub-tour of one parent and maintaining the relative sequence of genes of another parent. The second approach is a form of mutation, which is a genetic operator that reorders some of the gene values in a chromosome from their existing state. We apply heuristic and inversion mutation proposed by [43]. In the heuristic mutation, three genes are randomly selected from each parent, and all the possible combinations of the selected genes are generated. The best chromosome is considered as the offspring. In the inversion mutation, a sub-string of a parent chromosome is selected and used to produce an offspring.

3.3.3. Post improvement

GA returns the best set of chromosomes after any of the predefined stopping criteria is met. In the feasibility heuristic, an insertion of a CS is performed whenever a recharging is required. A route for an EV is defined as a sequence of targets and CSs $(d_0, t_i, t_j, d_n, \ldots, t_l, d_m, \ldots, d_0)$ where d_0 denotes the base depot, $i, j, l \in V$, and $m, n \in \bar{D}$. In some instances, it is possible to obtain a better fitness value by changing the position of the CSs in chromosomes. For example, authors in [44] proposed a dynamic programming-based approach to identify optimal location for intra-route facilities. Hence, to further enhance the quality of the GA solution, we propose a heuristic which is described by an example as follows: suppose that there are three CSs (d_1, d_2, and d_3), and the following string is a feasible path for an EV (the visited CSs are colored with red):

d_0	t_8	t_{11}	d_3	t_9	t_5	d_2	d_0

Here, a "sub-string" is a sequence of visits where the last node is a CS. We select the first sub-string and delete the CS from the sub-string. We then insert all the eligible CSs among the targets to generate new sub-routes. A CS is considered eligible if the EV starting from a target can reach that CS without visiting other CSs and also could reach the next target after recharging. New sub-routes are checked for battery capacity violation. The total distance traveled and the EV's remaining battery charge are stored for each feasible route. In the computational experiments, this considerably decreased the computational effort. The table (a) in Fig. 4 shows all the new sub-routes generated by CS insertion where the feasible sub-routes are highlighted. Then, we delete the infeasible sub-routes and add the second sub-string. The same CS insertion procedure for the second sub-string is repeated using the information stored in the previous step. The final possible routes are shown in the table (b) in Fig. 4.

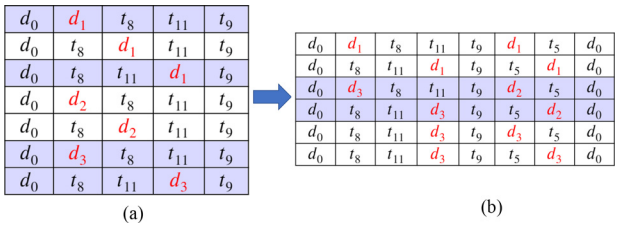

Fig. 4. Initial sub-tours generated in the first step of CS insertion heuristic (a), and final feasible routes (b).

The procedure is continued to visit the last node in all routes. Then the feasible route with the lowest distance is considered as the EV's total traveled distance. It should be noted that since the heuristic does not change the order of the targets, the chromosome remains the same. So, this heuristic is used only on final n best chromosomes returned by GA where n is a predefined user parameter (e.g., in this work we used n=15).

4. Experiments and results

In this section, we compare the computational performance of B&C algorithm and HHA. All the experiments were implemented in Python 3.7, and the MIP models were solved by Gurobi 9.0 [36] using a computer with an Intel ® Xeon ® CPU E5-2640, 2.60 GHz, and 80GB RAM.

4.1. Instance generation

For the computational experiments, we selected three data sets (A, B, and P) of the capacitated vehicle routing problem developed by Augerat et al. [45]. The data sets were modified to adapt for MEVRP by adding CSs. Additionally, a random data set was used to maintain the diversity in our experiments. The details of the data sets are as follows:

- *Random instances*: Random instances were generated in a square grid of size [100, 100], and the base depot at (50, 50). The number of CSs was set to five, and the locations of the depot as well as the CSs

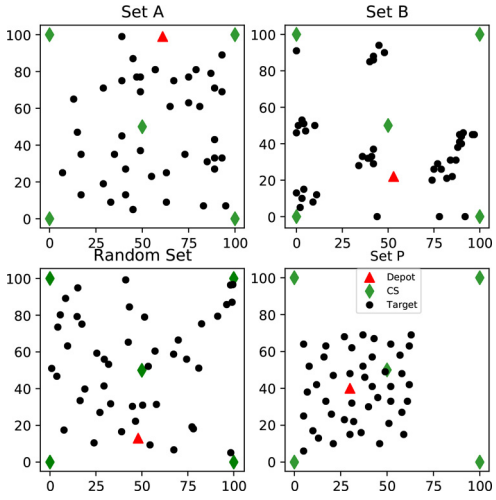

Fig. 5. Three data sets based on the study in [45] and a random set for computational experiments.

were fixed as apriori for all the random test instances. The number of targets varies from 10 to 50 in steps of five, while their locations were uniformly distributed within the square grid. For each of the generated instances, the number of EVs in the base depot is varied from two to eight. The battery capacity of EVs was set to 100, and the energy consumption rate is considered to be 0.8.

- *Augerat* et al.*instances*: The data sets reported in the study by Augerat et al. [45] are developed for capacitated vehicle routing problems. These instances include three sets which differ in distribution and number of targets, vehicle's capacity, demand/capacity, and number of vehicles. To make the instances compatible to MEVRP, we added five CSs. The coordination of CSs are similar for all instances within each data set. Also, the capacity of vehicles is considered as the battery capacity of EVs and capacity tightness as the fuel consumption rate in our problem. Fig. 5 shows the four different data sets.

4.2. Parameter tuning

Prior to the numerical experiments, we conducted an analysis on the parameters of maximum number of non-improving iterations for insertion ($I_i^{\max}$), maximum number of non-improving iterations for swaps ($I_s^{\max}$), GA population size (P_{size}), crossover rate (C_r) and mutation rate (M_r). A three-level Taguchi design [46] framework was used by considering three levels for each parameter using Minitab 19 [47]. A randomly chosen instance with 32 targets and five EVs was used for the experiments. For each combination of levels, HHA is run four times and the average of solutions for each combination is taken as response. The response values are then used as inputs, and the optimal levels of parameters are obtained. To demonstrate the performance of HHA algorithm for different values of parameters, additional computational experiments to investigate the impact of $I_i^{\max}$ and $I_s^{\max}$ parameters were performed. Random instances from each of the four sets were run for different values of

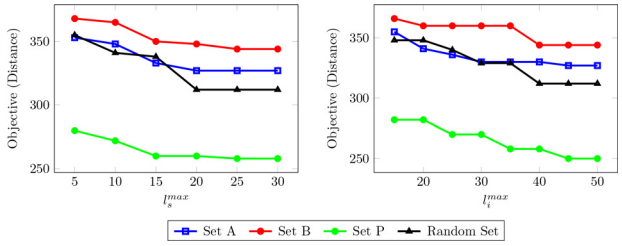

Fig. 6. The effect of $I_i^{\max}$ and $I_s^{\max}$ parameters on the objective of MEVRP.

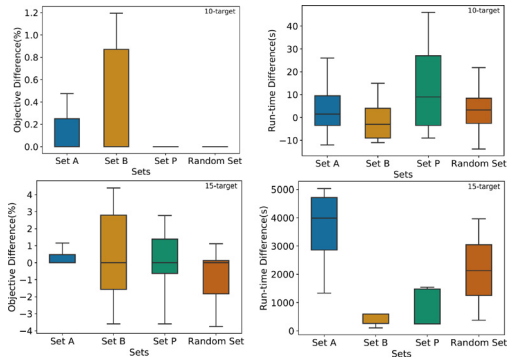

Fig. 7. Performance of the HHA compared to the B&C procedure for benchmark instances with 10 and 15 targets.

parameters. The tuned parameters are as follows: $I_i^{\max} = 40$, $I_s^{\max} = 25$, $P_{size} = 55$, $C_r = 0.6$ and $C_r = 0.2$

4.3. Experiment on MEVRP instances

4.3.1. Benchmark instances

For benchmark instances, a subset of targets were used which are the first 10 and 15 targets with two and three EVs within each data set, respectively. We solved the benchmark instances with HHA and compared the it with optimal or near-optimal solution solved by the B&C algorithm. Fig. 7 represents the differences in the objective function between HHA and B&C for the 10-targets and 15-targets instances. The objective difference in percentage and run-time in seconds are calculated as $\frac{Bamp;C(O)-HHA(O)}{Bamp;C(O)} \times 100$, and (B&C(t)-HHA(t)), where B&C(O) and HHA(O) indicate objective values, and B&C(t) and HHA(t) indicate the run-times of B&C and HHA, respectively. In total, 71 10-targets and 71 15-targets benchmark instances were solved. For all the 10-targets instances, the B&C algorithm was able to reach solutions within 1% of the optimality gap. For many of the 15-targets instances from sets A, B, and Random, the B&C algorithm was not able to find the optimal solution within the stipulated time limit of two hours. However, B&C found the optimal solution for most of the 15-targets instances from set P. Fig. 7 represents the run-time and objective function differences between HHA and B&C. For all the completed runs using B&C, HHA reached the optimal solution or near-optimal solution within less than 2.5% gap. For the instances where B&C was not able to reach optimality, the HHA approach could find a solution closer or better compared to the upper bound provided by B&C. In terms of run-time, for more than half of the 10-targets instances, HHA outperformed B&C. Also, HHA outperformed B&C with a much better run-time in the experiments with 15-targets instances. It should be noted that we removed the instances which reached the time-limit in Fig 7 using B&C to better illustrate the run-time difference in 15-targets instances. he characteristics of the instances and the detailed performance of algorithms are provided in Appendix A. The results from the benchmark instances indicate that the HHA approach is capable of providing highly efficient routes for EVs for smaller instances as well.

4.3.2. Large-scale instances

To demonstrate the performance of HHA on large scale problem sets, we considered 72 instances with the number of targets and EVs up to 60 and 15, respectively. Among these instances, 18 instances are from set A, and 17, 12, and 25 instances are from sets B, P, and Random, respectively. In 31 of the 72 total instances, B&C procedure was not able to find any feasible solution whereas HHA was capable of providing a feasible solution for all instances in just a few seconds. Table 2 represents the comparison between the B&C procedure and HHA for each set. The first column indicates the total number of instances, and the columns '# of Inf(B&C)' and '# of Inf(HHA)' indicate the number of instances in which the B&C procedure and HHA could not find any feasible

Table 1
Characteristics of the large-scale instances for different data sets .

Sets	Target range	Vehicles range	Consumption rate range	Battery Capacity range
A	[31,60]	[5,9]	[0.81,0.99]	100
B	[30,60]	[4,9]	[0.82,1]	100
P	[21,59]	[5,15]	[0.88,0.99]	[70,170]
Random	[20,50]	[4,10]	[0.8,0.8]	100

Table 2
Comparison of the performances of HHA and B&C procedure for large-scale instances .

Sets	# of Instances	# of Inf(B&C)	# of Inf(HHA)	AG-B&C(%)	ART-HHA(s)	AOD(%)
A	18	9	0	51	337	12.6
B	17	13	2	62.7	320	13.3
P	12	2	0	51.7	254	22.2
Random	25	7	0	60.8	263	13.6

solution in the stipulated time limit. The next two columns labeled 'AG-B&C(%)' and 'ART-HHA(%)' specify the average gap percentage and the average run-time in seconds obtained by the B&C and HHA approaches. The last column labeled 'AOD(%)' is the average objective difference percentage between B&C procedure and HHA. For example, in Table 2, for the 18 instances in set A, B&C algorithm found a feasible solution for nine instances with an average optimality gap of 51%, while HHA found a feasible solution for all 18 instances. Similarly, for sets B, P, and Random, B&C procedure found a feasible solution for 4, 10 and 18 instances with an average gap of 63.7%, 51.7%, and 60.8%, respectively. For all the experiments using large-scale instances, HHA outperformed B&C procedure in terms of computational time and the quality of the solution. Fig. 8 illustrates the performance of HHA compared to the B&C algorithm where the horizontal and vertical axes indicate the number of targets and the objective for different instances. Also, the disconnected lines indicate the instances where the B&C could not find any feasible solutions within the time limit.

4.4. Sensitivity analysis

4.4.1. Effect of the number of EVs

We analyze the effect of increasing the number of EVs on objective function and total distance traveled by all the EVs. As shown in Fig. 9, when there is an increase in the number of EVs, in general, the maximum distance traveled by the EVs decreases. However, the total distance traveled by all the EVs increases. The instance from set B has the highest increase in the total distance traveled by the EVs as the number of EVs is increased. This result is likely due to the cluster-like distribution of the targets that occurred in that data set. When the number of EVs surpasses the number of clusters, each cluster is assigned to more than one EV which increase the total distance. Another factor that may significantly affect the total distance is the position of the depot. In the cases where the depot is far from the targets, dispatching multiple EVs could result in a longer total distance. Since the targets are closer to each other in set P, an increase in the number of EVs would not result in a significant decrease or increase in the min-max or total distance, respectively. In conclusion, decision makers should evaluate the trade-off between the maximum and total distances while considering the distribution of the targets and locations of the depots.

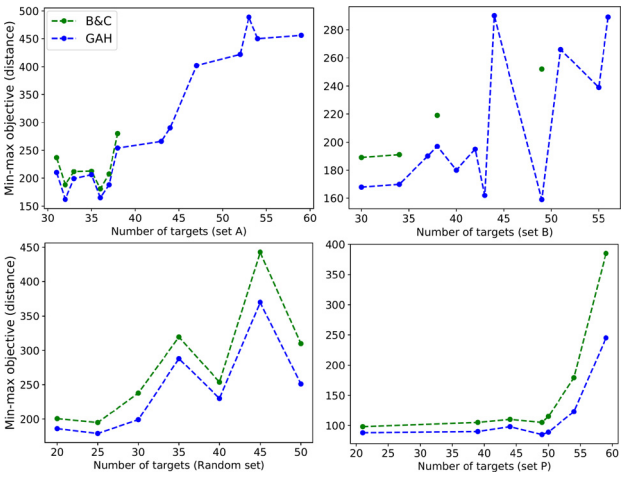

Fig. 8. Comparison between the performance of HHA and B&C for different large-scale instances.

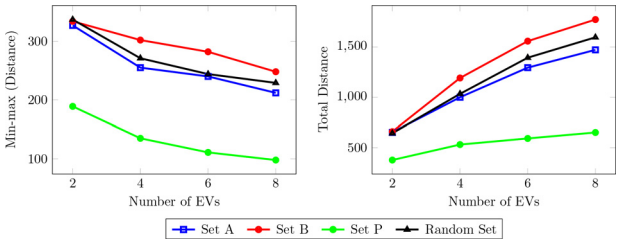

Fig. 9. Effect of increase in the number of EVs on the min-max and the total distance traveled by the EVs.

4.4.2. Effect of the number of charging stations

Another perspective is the impact of increase in the number of CSs on the min-max objective function. We chose an instance from each of the four data sets, and for each instance, we considered two to seven randomly located CSs. Fig. 10 shows the effect of number of CSs on MEVRP for different instances. An increase in the number of CSs has a lower impact on sets P and B sets where the targets are confined to a relatively smaller area. Due to the scattered distribution of the targets in the random set and set A, we observe a rapid decrease in the maximum

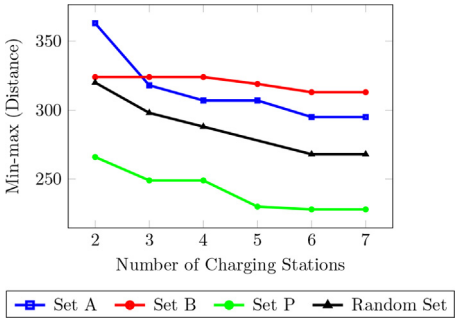

Fig. 10. Effect of increase in the number of charging stations on sets A, B, P and Random.

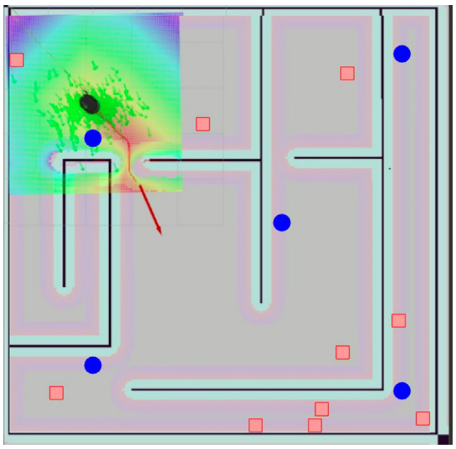

Fig. 11. Rviz navigation environment illustrated using Turtlebot package.

distance as the number of CSs decreases. This sort of analysis will help the logistics companies to evaluate the marginal benefits of adding more CSs based on the distribution of the customers in their network.

4.5. Data-driven simulation

Finally, we performed a data-driven simulation study using the Rviz [48] 3D-simulation environment for the robotic operating system (ROS) [49] middleware to showcase the application of AEVRP for robots. We used Turtlebots [50] from the turtlebot_stage package environment as shown in Fig. 11. The black oval indicates the base depot where the Turtlebots are stationed. The red rectangles indicate targets, and the blue circles represent the charging locations. The highlighted black lines are walls (obstacles) that prevent Turtlebots from moving from one target to another directly. The ROS navigation stack computes the distance between any pair of targets and depots in the environment. However, it is not possible to use the distance matrix in the AEVRP model since it does not satisfy the triangle inequality. Hence, to preserve the triangle inequality, we converted the distance between each pair of nodes calculated by ROS denoted as f'_{ij} to f_{ij} using Dijkstra's algorithm [40]. The Dijkstra algorithm creates a mapping between f'_{ij} to f_{ij} by calculating the shortest path between nodes. We use the new distance matrix as the cost to the AEVRP problem and conducted experiments with four different instances. In each instance, there are two Turtlebots, nine targets, and five charging locations. The experiments were performed using ROS Kinetic Kame [51] using a Ubuntu 16.04 (Xenial) release and the ROS publishers were developed using the Python programming language. Fig. 12 shows the traveled distance by each turtlebot for min-max and min-sum AEVRP with two and three turtlebots. In the instances 1 and 2 with two turtlebots, we observe the total distance traveled by Turtlebots are same, however, a significant difference between the traveled distance for each Turtlebot in the min-max versus min-sum. In instances 3 and 4, we notice a slight increase in the total traveled distance in min-max but considerable improvement in the workload balancing between the two Turtlebots. A similar pattern is observed in the experiment with three Turtlebots. In summary, on average, we observed a 2% increase in the total traveled distance by the Turtlebots using the min-max, however a 95% improvement in balancing the traveled distance between the Turtlebots while compared to the min-sum AEVRP.

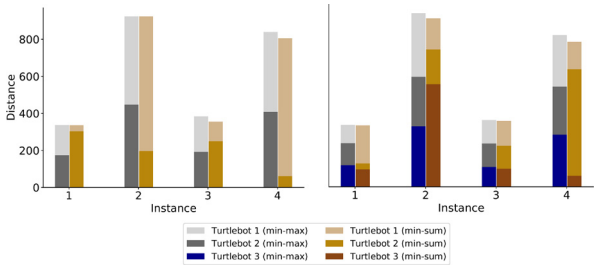

Fig. 12. Comparison among four different instances for total distance and workload balance for two (left), and three (right) turtlebots using min-max and min-sum approaches.

5. Managerial implication

The managerial implications of the min-max electric vehicle routing problem are significant for companies and organizations operating electric vehicle fleets. By addressing the limited range and battery degradation challenges, efficient routing and recharging strategies can be implemented to optimize the performance and utilization of EVs. This can lead to several managerial benefits:

1) Cost Reduction: MEVRP enables the minimization of the maximum distance traveled by any EV, which can result in cost savings by optimizing route planning and reducing overall transportation expenses.

2) Enhanced Efficiency: By incorporating charging stations and considering their availability in the routing strategy, MEVRP helps optimize recharging schedules and minimize EV downtime, thereby improving operational efficiency and productivity.

3) Environmental Sustainability: MEVRP supports the adoption and utilization of EVs, which contribute to reduced greenhouse gas emissions and promote a more sustainable transportation system. This aligns with corporate social responsibility goals and environmental regulations.

4) Improved Customer Service: Efficient routing strategies derived from MEVRP can lead to improved delivery or transportation services by minimizing delays, ensuring timely arrivals, and enhancing overall customer satisfaction.

5) Long-Term Fleet Planning: MEVRP can assist in making informed decisions regarding fleet expansion, vehicle acquisition, and charging infrastructure development, based on the optimization results and insights gained from the routing problem.

Overall, the application of MEVRP can have positive managerial implications by optimizing EV fleet operations, reducing costs, improving efficiency, promoting sustainability, and enhancing customer service in the transportation and logistics sector.

6. Conclusion

In this research, we consider a min-max routing problem for a fleet of EVs in the presence of charging stations. We proposed an efficient mixed-integer programming formulation and a hybrid heuristic algorithm with an embedded feasibility method to solve small and large-scale instances, respectively. Numerical studies are performed on randomly generated data sets and capacitated vehicle routing problem data sets with some modifications from literature. The efficacy of the proposed methods are benchmarked using extensive computational experiments. The results indicate that HHA was able to find solutions within 1% of the optimality gap for the 10-targets, and equal or better solutions in the 15-targets benchmark instances. Also, HHA was able to produce high-quality solutions in large-scale instances whereas the branch-and-cut procedure could not even find a feasible solution for most of the instances within the two hours time limit. The proposed methods are capable for facilitating routing and charging decisions for a fleet of EVs employed by logistics and transportation companies. The sensitivity analysis indicates that with a reasonable sacrifice in the total distance, the proposed approach can significantly decrease the travel time to visit some of the targets. In addition, this may also decrease the maintenance cost of the EVs due to fair and equitable distribution of workload among the EVs fleet. This could potentially decrease the aging or degradation of batteries in EVs.Furthermore, we conducted a data-driven simulation using the ROS package to investigate the application of the AEVRP for robots. Future work could include EVs with freight capacities and the

targets with time windows. Another extension could be incorporating uncertainties or non-linearity in the battery consumption rate as well as the location of target locations, especially for surveillance applications. From an algorithmic perspective, use of decomposition algorithms like the column generation method can also be investigated.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors wish to acknowledge the technical and financial support of the Automotive Research Center (ARC) in accordance with Cooperative Agreement W56HZV-19-2-0001 U.S. Army CCDC Ground Vehicle Systems Center (GVSC) Warren, MI.

Appendix A. Detailed Computational Results

Table (3) compares the Branch and Cut (B&C) algorithm with the Hybrid Heuristic Algorithm (HHA) for the random benchmark instances. The columns 2 to 5 show the number of targets, number of EVs, battery capacity (F), and energy consumption rate (K), respectively. The next four columns specify the performance of branch and cut algorithm with 'Obj,' 'Time(s),' 'Gap(%)' and 'Cuts(#)' indicating the objective function, run-time (seconds), the gap percentage, and the total number of connectivity cuts within the stipulated time limit, respectively. The last three columns indicate the performance of HHA algorithm with 'Obj','Time(s)', and 'PI(%)' representing the objective function, run-time (seconds) and the post improvement percentage, respectively. The PI(%) indicates the improvement obtained by using charging station insertion heuristic comparing to the best results of the HHA. Similarly, Tables (4),(5), and (6) compare the two methods over the Augerat instances. The results for large-scale instances can be found in Tables (7), (8), (9) and (10).

Table 3
Computational results of benchmark instances; set random .

Instance	Targets(#)	EV(#)	F	K	B&C				HHA		
					Obj	Time(s)	Gap(%)	Cuts(#)	Obj	Time(s)	PI(%)
c15-1-F1	10	2	100	0.8	279.7	68	0	101	279.7	22	0
	15	3	100	0.8	279.4	7200	29.3	1206	269	39	2
c15-2-F1	10	2	100	0.8	230.5	19	0	53	230.5	24	0
	15	3	100	0.8	220.1	7200	21.9	261	216	40	5
c15-3-F1	10	2	100	0.8	205.8	22	0	77	205.8	21	0
	15	3	100	0.8	190.7	7200	21.5	158	189	41	3
c15-4-F1	10	2	100	0.8	214.7	21	0	67	214	25	2
	15	3	100	0.8	205.7	7200	14.2	320	202	44	8
c15-5-F1	10	2	100	0.8	217.7	11	0	41	219	44	0
	15	3	100	0.8	191.9	7200	0.006	158	194	31	0
c20-1-F1	10	2	100	0.8	217.2	18	0	31	217.2	20	0
	15	3	100	0.8	214.9	7200	6	211	214.9	34	3
c20-2-F1	10	2	100	0.8	241.5	34	0	73	24.5	21	0
	15	3	100	0.8	211.6	7200	16.8	234	206	33	6
c20-3-F1	10	2	100	0.8	245.9	21	0	87	245.9	18	7
	15	3	100	0.8	212.4	7200	22	417	209	41	2
c20-4-F1	10	2	100	0.8	203	16	0	34	205	17	0
	15	3	100	0.8	180.8	7200	9	85	182	32	3
c20-5-F1	10	2	100	0.8	215	5.2	0	47	215	25	3
	15	3	100	0.8	212.9	7200	0.03	721	212.9	38	6
c25-1-F1	10	2	100	0.8	207.3	11	0	38	207.3	19	0
	15	3	100	0.8	197.1	2175	0	68	197.1	19	4
c25-2-F1	10	2	100	0.8	279.8	184	0	236	28	18	2
	15	3	100	0.8	264.8	7200	29.1	1285	258.6	35	0
c25-3-F1	10	2	100	0.8	236.5	39	0	83	236.5	29	0
	15	3	100	0.8	192.4	7200	11.1	129	192.4	35	5
c25-4-F1	10	2	100	0.8	237.9	35	0	144	237.9	26	0
	15	3	100	0.8	226.5	7200	23.9	1579	220	50	0
c25-5-F1	10	2	100	0.8	193.1	5	0	30	193.1	22	0
	15	3	100	0.8	191.3	7200	7	107	191.3	49	7
c30-1-F1	10	2	100	0.8	225.4	15	0	49	225.4	23	8
	15	3	100	0.8	228.5	7200	2	218	230	40	0
c30-2-F1	10	2	100	0.8	185	13	0	73	185	29	0
	15	3	100	0.8	159.3	423	0	92	159.3	47	5
c30-3-F1	10	2	100	0.8	203.9	11	0	60	203.9	31	0
	15	3	100	0.8	194.1	4009	0	152	194.1	47	3
c30-3-F1	10	2	100	0.8	227.3	11	0	61	227.3	27	0
	15	3	100	0.8	219.8	7200	7	166	221	59	0
c30-4-F1	10	2	100	0.8	227.3	11	0	61	227.3	27	4
	15	3	100	0.8	219.8	7200	7	166	221	59	0
c30-5-F1	10	2	100	0.8	230.4	10	0	50	230.4	13	3
	15	3	100	0.8	190.8	2170	0	138	193	41	6

Table 4
Computational results of benchmark instances; set A.

Instance	Targets(#)	EV(#)	F	K	B&C				HHA		
					Obj	Time(s)	Gap(%)	Cuts(#)	Obj	Time(s)	PI(%)
A-32-k5	10	2	100	0.82	275.6	20	0	53	275.6	23	0
	15	3	100	0.82	237.4	7200	12.8	111	237.4	55	4
A-n33-k5	10	2	100	0.89	237.9	26	0	68	239	28	0
	15	3	100	0.89	187	7200	10	66	185	53	3
A-n33-k6	10	2	100	0.90	151.4	24	0	70	151.4	29	2
	15	3	100	0.90	144	7200	11.3	116	144	58	0
A-n34-k5	10	2	100	0.92	230.3	35	0	49	230.3	75	1
	15	3	100	0.92	191.5	7200	12	94	188	73	2
A-n36-k5	10	2	100	0.88	244.4	17	0	67	244.4	37	3
	15	3	100	0.88	235.7	7200	19.3	323	232	74	4
A-n37-k5	10	2	100	0.81	194.6	7	0	27	194.6	29	0
	15	3	100	0.81	184	7200	9.4	226	184	67	0
A-n37-k6	10	2	100	0.95	234	17	0	83	237	33	0
	15	3	100	0.95	234	7200	12.7	360	234	45	0
A-n38-k5	10	2	100	0.96	213.4	5	0	31	213.4	29	4
	15	3	100	0.96	213.4	7200	18	142	213.2	45	8
A-n39-k5	10	2	100	0.95	197.7	22	0	40	199	35	0
	15	3	100	0.95	195.9	7200	25	177	190	54	6
A-n39-k6	10	2	100	0.88	223.1	24	0	102	225	47	0
	15	3	100	0.88	188	4673	0	161	192	62	0
A-n44-k6	10	2	100	0.95	266.9	24	0	114	266.9	51	0
	15	3	100	0.95	230	1400	0	177	234	71	5
A-n45-k6	10	2	100	0.99	243.3	15	0	55	243.3	37	0
	15	3	100	0.99	228.4	7200	1.7	162	228.4	55	7
A-n45-k7	10	2	100	0.91	236.8	9	0	45	236.8	34	2
	15	3	100	0.91	215.8	7200	1.2	211	220	69	2
A-n48-k7	10	2	100	0.89	262.7	16	0	70	262.7	38	0
	15	3	100	0.89	243.1	7200	20	302	239	81	7
A-n53-k7	10	2	100	0.95	189.9	4	0	32	192	35	0
	15	3	100	0.95	207.6	3438	0	129	211	77	3
A-n54-k7	10	2	100	0.96	246.8	14	0	82	246.8	38	2
	15	3	100	0.96	230.5	7200	20	780	228	44	5
A-n55-k9	10	2	100	0.93	203.2	7	0	47	203.2	39	0
	15	3	100	0.93	195.1	7200	2	205	195.1	51	0
A-n60-k9	10	2	100	0.92	255.5	13	0	55	255.5	47	0
	15	3	100	0.92	246.5	5091	0	270	246.5	92	8

Table 5
Computational results of benchmark instances; set B .

Instance	Targets(#)	EV(#)	F	K	B&C				HHA		
					Obj	Time(s)	Gap(%)	Cuts(#)	Obj	Time(s)	PI(%)
B-n31-k5	10	2	100	0.82	188.8	0	2	35	188.8	22	0
	15	3	100	0.82	236.6	7200	10	69	236.6	71	0
B-n34-k5	10	2	100	0.91	195.9	0	16	19	195.9	23	0
	15	3	100	0.91	195.2	7200	19.3	35	188.2	66	0
B-n35-k5	10	2	100	0.87	282.4	11	0	59	282.4	28	2
	15	3	100	0.87	259.7	7200	17.2	183	259.7	88	5
B-n38-k6	10	2	100	0.85	226.2	6	0	67	228	31	0
	15	3	100	0.85	173.3	241	0	144	173.3	80	5
B-n39-k5	10	2	100	0.88	194.3	6	0	45	194.3	55	0
	15	3	100	0.88	176.4	7200	1.3	87	179	72	2
B-n41-k6	10	2	100	0.95	243.3	8	0	36	243.3	37	0
	15	3	100	0.95	217.4	7200	14.1	163	216.8	72	5
B-n43-k6	10	2	100	0.87	184.4	99	0	49	184.4	67	4
	15	3	100	0.87	174.9	7200	11.8	212	175	89	7
B-n44-k7	10	2	100	0.92	210.9	9	0	34	210.9	29	0
	15	3	100	0.92	206.1	7200	16.5	105	203.8	91	3
B-n45-k5	10	2	100	0.97	190.3	4	0	20	190.3	35	1
	15	3	100	0.97	185.4	7200	11.4	135	183.6	82	3
B-n45-k6	10	2	100	0.99	177	13	0	22	177	47	3
	15	3	100	0.99	176.7	7200	11	135	178.5	88	0
B-n50-k7	10	2	100	0.87	199.9	4	0	37	199.9	23	0
	15	3	100	0.87	172.5	7200	11.7	25	168.2	90	0
B-n50-k8	10	2	100	0.92	284.6	34	0	210	288	32	0
	15	3	100	0.92	247.9	7200	19.9	897	242	72	4
B-n51-k7	10	2	100	0.98	248.2	15	0	31	248.2	26	2
	15	3	100	0.98	239.8	7200	34.6	64	225	68	6

(*continued on next page*)

Table 5 (*continued*)

Instance	Targets(#)	EV(#)	F	K	B&C				HHA		
					Obj	Time(s)	Gap(%)	Cuts(#)	Obj	Time(s)	PI(%)
B-n52-k7	10	2	100	0.87	209.1	5	0	20	209.1	31	3
	15	3	100	0.87	211.8	7200	10.7	106	210.6	72	5
B-n56-k7	10	2	100	0.88	197.1	5	0	43	197.1	35	0
	15	3	100	0.88	196.3	7200	16.8	79	194	74	6
B-n57-k7	10	2	100	1	298.4	32	0	119	301	33	3
	15	3	100	1	219.1	7200	4	139	222	63	0
B-n57-k9	10	2	100	0.89	269.2	20	0	61	269.2	44	0
	15	3	100	0.89	251.8	7200	29.1	467	248	69	4

Table 6
Computational results of benchmark instances; set P .

Instance	Targets(#)	EV(#)	F	K	B&C				HHA		
					Obj	Time(s)	Gap(%)	Cuts(#)	Obj	Time(s)	PI(%)
P-n19-k2	18	2	160	0.97	112.7	399	0	15	112.7	33	0
	18	3	160	0.97	89.5	4661	0	33	89.5	56	2
P-n20-k2	19	2	160	0.97	112.7	274	0	56	112.7	41	0
	19	3	160	0.97	90.1	7200	4.6	48	90.1	82	0
P-n21-k2	20	2	160	0.93	112.7	793	0	47	112.7	63	0
	20	3	160	0.93	90.1	7200	7.3	64	90.1	91	2
P-n22-k2	21	2	160	0.96	112.7	422	0	22	112.7	72	0
	21	3	160	0.96	90.9	7200	12.6	109	90.9	88	3
B-n22-k8	21	2	3000	0.94	158.7	0	0	50	158.7	91	2
	21	3	3000	0.94	115.9	7200	7.7	16	115	77	7
P-n40-k5	15	2	140	0.88	124.7	8	0	18	124.7	57	0
	15	3	140	0.88	94.6	675	0	41	96.2	66	6
P-n45-k5	15	2	150	0.92	124.7	15	0	9	124.7	44	1
	15	3	150	0.92	94.6	670	0	16	94.6	88	3
P-n50-k7	15	2	150	0.91	112.4	6	0	6	112.4	40	4
	15	3	150	0.91	86.9	271	0	21	86.9	53	0
P-n50-k8	15	2	120	0.99	112.4	8	0	19	112.4	58	3
	15	3	120	0.99	86.9	204	0	20	86.9	69	4
P-n50-k10	15	2	100	0.95	118.6	30	0	27	118.6	55	2
	15	3	100	0.95	86.9	602	0	79	86.9	80	5
P-n51-k10	15	2	80	0.97	153.3	4971	0	11	153.3	64	7
	15	3	80	0.97	100.7	80	0	17	102	77	6
P-n55-k7	15	2	170	0.88	112.4	7	0	3	112.4	68	0
	15	3	170	0.88	86.9	254	0	27	86.9	76	0
P-n55-k10	15	2	115	0.91	112.4	10	0	12	112.4	71	2
	15	3	115	0.91	86.9	188	0	58	86.9	81	6
P-n55-k15	15	2	70	0.99	129.4	128	0	20	129.4	58	4
	15	3	70	0.99	109.5	7200	17.3	68	107.2	87	4
P-n60-k10	15	2	120	0.95	112.4	6	0	28	112.4	45	0
	15	3	120	0.95	86.9	708	0	30	86.9	79	2
P-n60-k15	15	2	130	0.94	129.4	349	0	7	129.4	66	1
	15	3	130	0.94	92.4	1614	0	21	92.4	93	5

Table 7
Computational results of Large-scale instances; set Random .

Instance	Targets(#)	EV(#)	F	K	B&C			HHA	
					Obj	Time(s)	Gap(%)	Obj	Time(s)
c25-1-F1	25	5	100	0.8	178	7200	35	152	129
c25-2-F1	25	5	100	0.8	254	7200	56	201	191
c25-3-F1	25	5	100	0.8	182	7200	38	159	155
c25-4-F1	25	5	100	0.8	257	7200	55	199	125
c25-5-F1	25	5	100	0.8	155	7200	28	144	118
c35-1-F1	35	7	100	0.8	-	7200	-	266	190
c35-2-F1	35	7	100	0.8	277	7200	58	243	199
c35-3-F1	35	7	100	0.8	235	7200	52	201	206
c35-4-F1	35	7	100	0.8	260	7200	60	238	203
c35-5-F1	35	7	100	0.8	177	7200	44	160	218
c40-1-F1	40	8	100	0.8	209	7200	51	176	228

(*continued on next page*)

Table 7 (*continued*)

Instance	Targets(#)	EV(#)	F	K	B&C			HHA	
					Obj	Time(s)	Gap(%)	Obj	Time(s)
c40-2-F1	40	8	100	0.8	-	7200	-	252	265
c40-3-F1	40	8	100	0.8	301	7200	70	269	277
c40-4-F1	40	8	100	0.8	249	7200	63	223	279
c40-5-F1	40	8	100	0.8	308	7200	65	287	258
c45-1-F1	45	9	100	0.8	-	7200	-	277	301
c45-2-F1	45	9	100	0.8	252	7200	66	236	288
c45-3-F1	45	9	100	0.8	439	7200	98	361	331
c45-4-F1	45	9	100	0.8	483	7200	98	390	360
c45-5-F1	45	9	100	0.8	383	7200	77	342	327
c50-1-F1	50	10	100	0.8	-	7200	-	282	301
c50-2-F1	50	10	100	0.8	329	7200	81	276	344
c50-3-F1	50	10	100	0.8	-	7200	-	321	431
c50-4-F1	50	10	100	0.8	-	7200	-	330	460
c50-5-F1	50	10	100	0.8	-	7200	-	289	397

Table 8
Computational results of large-scale instances; set A .

Instance	Targets(#)	EV(#)	F	K	B&C			HHA	
					Obj	Time(s)	Gap(%)	Obj	Time(s)
A-32-k5	31	5	100	0.82	237	7200	47	212	183
A-n33-k5	32	5	100	0.89	188	7200	42	162	224
A-n33-k6	32	6	100	0.9	186	7200	54	141	241
A-n34-k5	33	5	100	0.92	228	7200	43	201	225
A-n36-k5	35	5	100	0.88	246	7200	51	213	244
A-n37-k5	36	5	100	0.81	199	7200	42	178	239
A-n37-k6	36	6	100	0.95	-	7200	-	166	305
A-n38-k5	37	5	100	0.96	225	7200	48	208	298
A-n39-k5	38	5	100	0.95	354	7200	64	332	312
A-n39-k6	38	6	100	0.88	329	7200	68	279	295
A-n44-k6	43	6	100	0.95	-	7200	-	298	320
A-n45-k6	44	6	100	0.99	-	7200	-	290	358
A-n45-k7	44	7	100	0.91	-	7200	-	288	408
A-n48-k7	47	7	100	0.89	-	7200	-	278	422
A-n53-k7	52	7	100	0.95	-	7200	-	320	396
A-n54-k7	53	7	100	0.96	-	7200	-	331	410
A-n55-k9	54	9	100	0.93	-	7200	-	326	572
A-n60-k9	59	9	100	0.92	-	7200	-	299	627

Table 9
Computational results of large-scale instances; set B .

Instance	Targets(#)	EV(#)	F	K	B&C			HHA	
					Obj	Time(s)	Gap(%)	Obj	Time(s)
B-n31-k5	30	5	100	0.82	189	7200	55	171	287
B-n34-k5	33	5	100	0.91	191	7200	51	162	214
B-n35-k5	34	5	100	0.87	-	7200	-	195	290
B-n38-k6	37	6	100	0.85	-	7200	-	180	305
B-n39-k5	38	5	100	0.88	219	7200	68	187	244
B-n41-k6	40	6	100	0.95	-	7200	-	278	280
B-n43-k6	42	6	100	0.87	-	7200	-	200	322
B-n44-k7	43	7	100	0.92	-	7200	-	220	233
B-n45-k5	44	5	100	0.97	-	7200	-	205	270
B-n45-k6	44	6	100	0.99	-	7200	-	215	325
B-n50-k7	49	7	100	0.87	252	7200	81	217	319
B-n50-k8	49	8	100	0.92	-	7200	-	261	418
B-n51-k7	50	7	100	0.98	-	7200	-	-	-
B-n52-k7	51	7	100	0.87	-	7200	-	266	403
B-n56-k7	55	7	100	0.88	-	7200	-	239	406
B-n57-k7	56	7	100	1	-	7200	-	-	-
B-n57-k9	56	9	100	0.89	-	7200	-	289	492

Table 10
Computational results of large-scale instances; set P .

Instance	Targets(#)	EV(#)	F	K	B&C			HHA	
					Obj	Time(s)	Gap(%)	Obj	Time(s)
P-n22-k8	21	8	3000	0.94	98	7200	46	82	154
P-n40-k5	39	5	150	0.88	105	7200	21	96	221
P-n45-k5	44	5	150	0.92	110	7200	18	107	215
P-n50-k7	49	7	120	0.91	105	7200	37	90	174
P-n50-k8	49	8	100	0.99	129	7200	53	88	250
P-n50-k10	49	10	80	0.95	111	7200	53	82	201
P-n51-k10	50	10	170	0.97	115	7200	54	89	20
P-n55-k7	54	7	115	0.88	292	7200	77	176	326
P-n55-k10	54	10	70	0.91	179	7200	70	124	249
P-n55-k15	54	15	120	0.99	-	7200	-	198	283
P-n60-k10	59	15	80	0.95	-	7200	-	242	309
P-n60-k15	59	15	130	0.95	385	7200	88	270	450

References

[1] H. Nazaripouya, B. Wang, D. Black, Electric vehicles and climate change: additional contribution and improved economic justification, IEEE Electrif. Mag. 7 (2) (2019) 33–39.
[2] S. Solaymani, Co2 emissions patterns in 7 top carbon emitter economies: the case of transport sector, Energy 168 (2019) 989–1001.
[3] M. Mahjoob, S.S. Fazeli, S. Milanlouei, A.K. Mohammadzadeh, L.S. Tavassoli, J.S. Noble, Green supply chain network design with emphasis on inventory decisions, Sustainable Operations and Computers 2 (2021) 214–229.
[4] J. Wu, H. Liao, J.-W. Wang, T. Chen, The role of environmental concern in the public acceptance of autonomous electric vehicles: a survey from china, Transp Res Part F Traffic Psychol Behav 60 (2019) 37–46.
[5] Z. Yi, J. Smart, M. Shirk, Energy impact evaluation for eco-routing and charging of autonomous electric vehicle fleet: ambient temperature consideration, Transp. Res. Part C Emerg. Technol. 89 (2018) 344–363.
[6] H. Zhang, C.J. Sheppard, T.E. Lipman, S.J. Moura, Joint fleet sizing and charging system planning for autonomous electric vehicles, IEEE trans Intell Transp Syst (2019).
[7] K. Sundar, S. Venkatachalam, S. Rathinam, An exact algorithm for a fuel-constrained autonomous vehicle path planning problem, arXiv preprint arXiv:1604.08464 (2016).
[8] M. Schneider, A. Stenger, D. Goeke, The electric vehicle-routing problem with time windows and recharging stations, Transportation Science 48 (4) (2014) 500–520.
[9] G. Hiermann, J. Puchinger, S. Ropke, R.F. Hartl, The electric fleet size and mix vehicle routing problem with time windows and recharging stations, Eur. J. Oper. Res. 252 (3) (2016) 995–1018.
[10] P. Slowik, N. Lutsey, The Continued Transition to Electric Vehicles in US Cities, 2018,
[11] S.S. Fazeli, S. Venkatachalam, R.B. Chinnam, A. Murat, Two-stage stochastic choice modeling approach for electric vehicle charging station network design in urban communities, IEEE trans Intell Transp Syst (2020).
[12] Ç. Koç, I. Karaoglan, The green vehicle routing problem: a heuristic based exact solution approach, Appl. Soft Comput. 39 (2016) 154–164.
[13] K. Sundar, S. Rathinam, Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots, IEEE T AUTOM SCI ENG 11 (1) (2013) 287–294.
[14] G. Desaulniers, F. Errico, S. Irnich, M. Schneider, Exact algorithms for electric vehicle-routing problems with time windows, Oper Res 64 (6) (2016) 1388–1405.
[15] F.Y. Vincent, A.P. Redi, Y.A. Hidayat, O.J. Wibowo, A simulated annealing heuristic for the hybrid vehicle routing problem, Appl. Soft Comput. 53 (2017) 119–132.
[16] A.A. Juan, J. Goentzel, T. Bektaş, Routing fleets with multiple driving ranges: is it possible to use greener fleet configurations? Appl. Soft Comput. 21 (2014) 84–94.
[17] M. Mahjoob, S.S. Fazeli, S. Milanlouei, L.S. Tavassoli, M. Mirmozaffari, A modified adaptive genetic algorithm for multi-product multi-period inventory routing problem, Sustainable Operations and Computers 3 (2022) 1–9.
[18] S. Erdoğan, E. Miller-Hooks, A green vehicle routing problem, Transportation research part E: logistics and transportation review 48 (1) (2012) 100–114.
[19] D. Levy, K. Sundar, S. Rathinam, Heuristics for routing heterogeneous unmanned vehicles with fuel constraints, Mathematical Problems in Engineering 2014 (2014).
[20] B. Lunz, Z. Yan, J.B. Gerschler, D.U. Sauer, Influence of plug-in hybrid electric vehicle charging strategies on charging and battery degradation costs, Energy Policy 46 (2012) 511–519.
[21] A.M. Campbell, D. Vandenbussche, W. Hermann, Routing for relief efforts, Transportation Science 42 (2) (2008) 127–145.
[22] M. Torabbeigi, G.J. Lim, S.J. Kim, Drone delivery scheduling optimization considering payload-induced battery consumption rates, J INTELL ROBOT SYST 97 (3) (2020) 471–487.
[23] S.J. Zaloga, Unmanned Aerial Vehicles: Robotic Air Warfare 1917–2007, Bloomsbury Publishing, 2011.
[24] S.G. Manyam, D.W. Casbeer, K. Sundar, Path planning for cooperative routing of air-ground vehicles, in: 2016 American Control Conference (ACC), IEEE, 2016, pp. 4630–4635.
[25] A.C. Kapoutsis, S.A. Chatzichristofis, E.B. Kosmatopoulos, Darp: divide areas algorithm for optimal multi-robot coverage path planning, J Intell Robat Syst 86 (3–4) (2017) 663–680.
[26] E. Yakıcı, A heuristic approach for solving a rich min-max vehicle routing problem with mixed fleet and mixed demand, CAIE 109 (2017) 288–294.
[27] J.F. Sze, S. Salhi, N. Wassan, The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: an effective hybridisation of adaptive variable neighbourhood search and large neighbourhood search, Transport Res B-METH 101 (2017) 162–184.
[28] K.V. Narasimha, E. Kivelevitch, B. Sharma, M. Kumar, An ant colony optimization technique for solving min–max multi-depot vehicle routing problem, Swarm Evol Comput 13 (2013) 63–73.
[29] X. Wang, B. Golden, E. Wasil, R. Zhang, The min–max split delivery multi-depot vehicle routing problem with minimum service time requirement, CAIE 71 (2016) 110–126.
[30] J. Carlsson, D. Ge, A. Subramaniam, A. Wu, Y. Ye, Solving min-max multi-depot vehicle routing problem, Lectures on global optimization 55 (2009) 31–46.
[31] C. Chakraborty, R. Roy, Markov decision process based optimal gateway selection algorithm, Int. Journal of Systems, Algorithms & Applications (IJSAA) 2 (2012) 48–52.
[32] A. Sarkar, M.Z. Khan, M.M. Singh, A. Noorwali, C. Chakraborty, S.K. Pani, Artificial neural synchronization using nature inspired whale optimization, IEEE Access 9 (2021) 16435–16447.
[33] S. Zhang, Y. Gajpal, S. Appadoo, A meta-heuristic for capacitated green vehicle routing problem, Ann. Oper. Res. 269 (1–2) (2018) 753–771.
[34] K. Sundar, S. Venkatachalam, S.G. Manyam, Path planning for multiple heterogeneous unmanned vehicles with uncertain service times, in: 2017 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, 2017, pp. 480–487.
[35] S. Venkatachalam, M. Bansal, J.M. Smereka, J. Lee, Two-stage stochastic programming approach for path planning problems under travel time and availability uncertainties, arXiv preprint arXiv:1910.04251 (2019).
[36] L. Gurobi Optimization, Gurobi optimizer reference manual, 2020, http://www.gurobi.com.
[37] V. Sirimuvva Chirala, K. Sundar, S. Venkatachalam, J.M. Smereka, S. Kassoumeh, Heuristics for multi-vehicle routing problem considering human-robot interactions, arXiv e-prints (2022) arXiv–2208.
[38] K. Helsgaun, An effective implementation of the lin–kernighan traveling salesman heuristic, Eur. J. Oper. Res. 126 (1) (2000) 106–130.
[39] S. Khuller, A. Malekian, J. Mestre, To fill or not to fill: the gas station problem, in: European Symposium on Algorithms, Springer, 2007, pp. 534–545.
[40] E.W. Dijkstra, et al., A note on two problems in connexion with graphs, Numerische mathematik 1 (1) (1959) 269–271.
[41] P. Hansen, N. Mladenović, J.A.M. Pérez, Variable neighbourhood search: methods and applications, Ann. Oper. Res. 175 (1) (2010) 367–407.
[42] L. Davis, Applying adaptive algorithms to epistatic domains, in: IJCAI, 85, 1985, pp. 162–164.
[43] M. Gen, R. Cheng, Genetic Algorithms and Manufacturing Systems Design, John Wiley & Sons, Inc., 1996.
[44] M. Schiffer, G. Walther, An adaptive large neighborhood search for the location-routing problem with intra-route facilities, Transportation Science 52 (2) (2018) 331–352.
[45] P. Augerat, J.M. Belenguer, E. Benavent, A. Corberán, D. Naddef, G. Rinaldi, Computational results with a branch and cut code for the capacitated vehicle routing problem, volume 34, IMAG, 1995.
[46] G. Taguchi, Introduction to quality engineering: designing quality into products and processes, Technical Report, 1986.
[47] Minitab 19 statistical software (2020). [computer software]. state college, pa: Minitab, inc, 2020, http://www.minitab.com.
[48] D. Hershberger, D. Gossow, J. Faust, Rviz, 3d visualization tool for ros, URL: http://wiki. ros. org/rviz [cited 06-12-2020]
[49] ROS, Robot operating system, URL: https://www.ros.org/ [cited 06-12-2020]
[50] P.R. kit, Turtlebot, URL: https://www.turtlebot.com/ [cited 06-12-2020]
[51] ROS, Ros kinetic kame, URL: http://wiki.ros.org/kinetic [cited 06-12-2020]